Gene Orthology Inference via Large-Scale Rearrangements for Partially Assembled Genomes

Diego P. Rubert
Faculdade de Computação, Universidade Federal de Mato Grosso do Sul, Campo Grande, MS, Brasil
Faculty of Technology and Center for Biotechnology (CeBiTec), Bielefeld University, Germany

Marília D. V. Braga[1]
Faculty of Technology and Center for Biotechnology (CeBiTec), Bielefeld University, Germany

Abstract

Recently we developed a gene orthology inference tool based on genome rearrangements (*Journal of Bioinformatics and Computational Biology* 19:6, 2021). Given a set of genomes our method first computes all pairwise gene similarities. Then it runs pairwise ILP comparisons to compute optimal gene matchings, which minimize, by taking the similarities into account, the weighted rearrangement distance between the analyzed genomes (a problem that is NP-hard). The gene matchings are then integrated into gene families in the final step. Although the ILP is quite efficient and could conceptually analyze genomes that are not completely assembled but split in several contigs, our tool failed in completing that task. The main reason is that each ILP pairwise comparison includes an optimal *capping* that connects each end of a linear segment of one genome to an end of a linear segment in the other genome, producing an exponential increase of the search space.

In this work, we design and implement a heuristic capping algorithm that replaces the optimal capping by clustering (based on their gene content intersections) the linear segments into $m \geq 1$ subsets, whose ends are capped independently. Furthermore, in each subset, instead of allowing all possible connections, we let only the ends of content-related segments be connected. Although there is no guarantee that m is much bigger than one, and with the possible side effect of resulting in sub-optimal instead of optimal gene matchings, the heuristic works very well in practice, from both the speed performance and the quality of computed solutions. Our experiments on real data show that we can now efficiently analyze fruit fly genomes with unfinished assemblies distributed in hundreds or even thousands of contigs, obtaining orthologies that are more similar to FlyBase orthologies when compared to orthologies computed by other inference tools. Moreover, for complete assemblies the version with heuristic capping reports orthologies that are very similar to the orthologies computed by the optimal version of our tool. Our approach is implemented into a pipeline incorporating the pre-computation of gene similarities.

2012 ACM Subject Classification Applied computing → Bioinformatics

Keywords and phrases Comparative genomics, double-cut-and-join, indels, gene orthology

Digital Object Identifier 10.4230/LIPIcs.WABI.2022.24

Supplementary Material Both the original version with optimal capping and the new modified version with heuristic capping can be downloaded from our GitLab server at:
Software (Source Code): `https://gitlab.ub.uni-bielefeld.de/gi/FFGC`

Acknowledgements We thank the anonymous reviewers for their valuable comments.

[1] Corresponding author

1 Introduction

The study of distances and parsimonious evolutionary scenarios based on large-scale genome rearrangements traditionally depends on the pre-computation of *gene families.* Computing such a distance is usually polynomial when genomes have at most one gene per family [4, 7, 13] or NP-hard otherwise [3, 5, 8, 20, 21]. These works adopt several rearrangement models and among the most popular ones is the double-cut-and-join (DCJ) operation [24], which mimics organizational rearrangements, such as inversions, fusions, fissions and translocations.

An alternative (NP-hard) *family-free* setting for genome rearrangement studies was proposed in 2013 [6] and further extended [16, 19], in a model that does not require the pre-computation of gene families and, besides DCJ operations, takes into account insertions and deletions of DNA segments, collectively called *indels.* This model is able to infer pairwise orthologs between two genomes directly, simultaneously based on gene similarities and rearrangements. In practice, its optimization function can be solved exactly due to an ILP formulation [19] that is called FF-DCJ-Indel and also reports an optimal matching of orthologs between the two analyzed genomes. (The ILP FF-DCJ-Indel is itself based on the previous formulations for family-based approaches [5, 21].)

With these achievements we were able to invert the traditional paradigm of genome rearrangement studies: instead of requiring the gene families to proceed with rearrangement comparisons, it became possible to use rearrangement comparisons for inferring the gene families[2]. Indeed, in our most recent work [18], we did a first attempt of using FF-DCJ-Indel for inferring genome-scale gene families across several species. More precisely, given a set of genomes, our method first computes all pairwise optimal gene matchings, which are integrated into gene families in the second step, resulting in a complete pipeline called DiffMGC, whose inferences displayed good quality in the analysis of completely assembled genomes.

However, although the integrated FF-DCJ-Indel is quite efficient and could conceptually analyze genomes that are not completely assembled but split in several contigs, it failed in completing that task. The main reason is that each ILP pairwise comparison includes an optimal *capping* that must allow the end of any linear segment of one genome to be matched to the end of any linear segment of the other genome. The optimal capping then produces an exponential increase of the search space.

In this work, we design and implement a heuristic capping algorithm that replaces the optimal capping by clustering (based on their gene content intersections) the linear segments into $m \geq 1$ subsets, so that the ends of the linear segments in the same subset S can only be matched to elements of S. Furthermore, in each subset, instead of allowing all possible connections, we let only the ends of content-related segments be connected. Although there is no guarantee that m is much bigger than one, and with the possible side effect of resulting in sub-optimal instead of optimal gene matchings, the heuristic works very well in practice, from both the speed performance and the quality of computed solutions.

We call DiffMGC$\overset{\approx}{\mathrm{H}}$ the new complete pipeline adopting the heuristic capping for FF-DCJ-Indel. Our experiments on real data show that we can now efficiently analyze fruit fly genomes with unfinished assemblies distributed in hundreds or even thousands of contigs.

[2] To be more precise, another attempt called MSOAR [22] was made before our studies, the differences being that MSOAR first infers gene families based on similarities and then computes a matching based on a heuristic including structural rearrangements and tandem duplications, while FF-DCJ-Indel takes similarities and rearrangements simultaneously into account for inferring an optimal matching, in a rearrangement model including DCJ and mimicking all content modifications with insertions and deletions of DNA segments. MSOAR was not maintained and is no longer operational, therefore we could never compare its performance to FF-DCJ-Indel.

We compared the gene families inferred by DiffMGC$\overset{\approx}{\text{H}}$ to Oma [10,17], ProteinOrtho [14], and Poff [15]. The orthologies inferred by DiffMGC$\overset{\approx}{\text{H}}$ are more similar to FlyBase orthologies when compared to orthologies computed by these other inference tools. Moreover, for complete assemblies DiffMGC$\overset{\approx}{\text{H}}$ reports orthologies that are very similar to the orthologies computed by DiffMGC, which is the optimal version of our tool.

2 Orthology inference via family-free genome rearrangements

For studying large-scale genome rearrangements a high-level view of a *chromosome* is adopted. In this view each chromosome is represented by a sequence of *genes*. Since each gene is an oriented DNA fragment, we need to distinguish its two possible representations: a gene g is represented by the symbol g itself, if it is read in direct orientation, or by the symbol $\overline{g}$, if it is read in reverse orientation. In our notation, all genes of a linear chromosome are concatenated in a string that can be read in any of the two directions and is flanked by square brackets. As an example, let $C = [\overline{6}\,1\,8\,9\,\overline{4}]$ be a linear chromosome. A *genome* is then a set of chromosomes and can be transformed with the following types of mutations:

1. **Structural rearrangements (DCJ operations):** A *cut* performed on a chromosome C of a genome $\mathbb{A}$ separates two adjacent genes of C. A *double-cut and join* or *DCJ* applied on genome $\mathbb{A}$ is the operation that performs cuts in two different positions of distinct chromosomes or of the same chromosome of $\mathbb{A}$, creating four open ends, and joins these open ends in a different way [4, 24]. For example, let $\mathbb{A} = \{[\overline{6}\,1\,8\,9\,\overline{4}], [3\,\overline{5}\,\overline{7}\,2]\}$, and consider a DCJ that cuts between genes 1 and 8 of its first chromosome and between genes 7 and 2 of its second chromosome, creating segments $\overline{6}\,1\bullet$, $\bullet 8\,9\,\overline{4}$, $3\,\overline{5}\,\overline{7}\bullet$ and $\bullet 2$ (where the symbols $\bullet$ represent the open ends). If we join the first with the fourth and the second with the third open end, we get $\mathbb{A}' = \{[\overline{6}\,1\,2], [3\,\overline{5}\,\overline{7}\,8\,9\,\overline{4}]\}$, that is, the described DCJ operation is a translocation transforming $\mathbb{A}$ into $\mathbb{A}'$. Indeed, a DCJ operation can correspond not only to a translocation but to several structural rearrangements, such as an inversion, a fusion or a fission.
2. **Content-modifying (indel operations):** The content of a chromosome can be modified with *insertions* and with *deletions* of blocks of contiguous genes, collectively called *indel* operations. Note that at most one chromosome can be entirely deleted or inserted at once. As an example, consider the deletion of segment $\overline{7}\,8\,9$ from chromosome $[3\,\overline{5}\,\overline{7}\,8\,9\,\overline{4}]$, resulting in chromosome $[3\,\overline{5}\,\overline{4}]$. A gene cannot be deleted and then reinserted, nor inserted and then deleted. This restriction prevents the *free lunch* artifact of sorting one genome into the other by simply deleting the chromosomes of the first and inserting the chromosomes of the second, ignoring their common parts.

2.1 Computing an optimal set of orthologs between two genomes

We can represent the pairwise similarities between the genes of genome $\mathbb{A}$ and the genes of genome $\mathbb{B}$ in the so called *gene similarity graph* [6], denoted by $\mathcal{S}(\mathbb{A}, \mathbb{B})$. This is a weighted bipartite graph that has a vertex for each gene in genome $\mathbb{A}$ and a vertex for each gene in genome $\mathbb{B}$. Furthermore, for each pair of genes $g_1 \in \mathbb{A}, g_2 \in \mathbb{B}$, denote by $\sigma(g_1, g_2)$ their *normalized similarity*, a value that ranges in the interval $[0, 1]$. Given a threshold $0 \leq x \leq 1$, if $\sigma(g_1, g_2) \geq x$ there is an edge e connecting g_1 and g_2 in $\mathcal{S}(\mathbb{A}, \mathbb{B})$ whose weight is $w(e) = \sigma(g_1, g_2)$. In addition, to each vertex u of $\mathcal{S}(\mathbb{A}, \mathbb{B})$ we assign a weight $w(u)$ that can be obtained as follows: $w(u) = \max\{\sigma(uv) \mid uv \in \mathcal{S}(A, B)\}$, that is, $w(u)$ is the maximum similarity among the edges incident to the vertex (or gene) u in $\mathcal{S}(\mathbb{A}, \mathbb{B})$.

A matching M from $\mathcal{S}(\mathbb{A},\mathbb{B})$, here also called an *ortholog-set*, defines the tuple $(\mathbb{A},\mathbb{B},M)$, in which every two genes a,b, such that $a \in \mathbb{A}$, $b \in \mathbb{B}$ and $ab \in M$, are considered to be *orthologs*. The *complement* of M, denoted by $\widetilde{M}$, is the set composed of genes whose corresponding vertices in $\mathcal{S}(\mathbb{A},\mathbb{B})$ are M-unsaturated.

The DCJ-indel distance $\mathrm{d}^{\mathrm{ID}}_{\mathrm{DCJ}}(\mathbb{A},\mathbb{B},M)$ is the minimum number of DCJ and indel operations required to transform $\mathbb{A}$ into $\mathbb{B}$ assuming the orthologs given by M and allowing only the genes belonging to the complement $\widetilde{M}$ to be inserted or deleted. It can be computed using an approach relying on the cycles and paths of a graph that represents the structural relation between genomes $\mathbb{A}$ and $\mathbb{B}$ according to the ortholog-set M [7,19] (this graph is equivalent to a consistent decomposition of the family-free relational graph, described in Section 2.2 and represented in Figure 1 (bottom)). Together with the weights of edges and vertices of $\mathcal{S}(\mathbb{A},\mathbb{B})$, the DCJ-indel distance $\mathrm{d}^{\mathrm{ID}}_{\mathrm{DCJ}}$ allows the computation of the weighted rearrangement distance $\mathrm{wd}^{\mathrm{ID}}_{\mathrm{DCJ}}$ [19]:

$$\mathrm{wd}^{\mathrm{ID}}_{\mathrm{DCJ}}(\mathbb{A},\mathbb{B},\mathcal{S},M) = \mathrm{d}^{\mathrm{ID}}_{\mathrm{DCJ}}(\mathbb{A},\mathbb{B},M) + |M| - w(M) + w(\widetilde{M}).$$

Then, given that $\mathfrak{M}$ is the set of all possible ortholog-sets in $\mathcal{S}(\mathbb{A},\mathbb{B})$, the rearrangement distance between $\mathbb{A}$ and $\mathbb{B}$ is the result of the following optimization:

$$\textsc{Diff}(\mathbb{A},\mathbb{B},\mathcal{S}) = \min_{M\in\mathfrak{M}} \{\mathrm{wd}^{\mathrm{ID}}_{\mathrm{DCJ}}(\mathbb{A},\mathbb{B},\mathcal{S},M)\}\ .$$

Figure 1 shows examples of ortholog-sets and their distances. Denote by $\textsc{Diff}\mathrm{M}(\mathbb{A},\mathbb{B},\mathcal{S})$ an *optimal* ortholog-set in $\mathcal{S}(\mathbb{A},\mathbb{B})$, which is an ortholog-set whose rearrangement distance equals $\textsc{Diff}(\mathbb{A},\mathbb{B},\mathcal{S})$. Computing the rearrangement distance $\textsc{Diff}(\mathbb{A},\mathbb{B},\mathcal{S})$ and finding an optimal ortholog-set $\textsc{Diff}\mathrm{M}(\mathbb{A},\mathbb{B},\mathcal{S})$ are NP-hard problems [19].

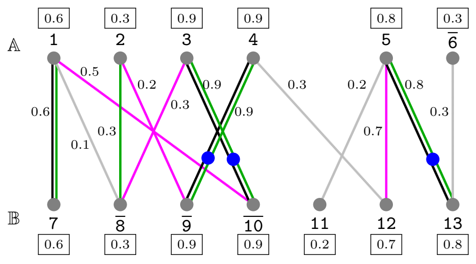
2.2 Family-free relational graph

One approach for solving the NP-hard problems $\textsc{Diff}(\mathbb{A},\mathbb{B},\mathcal{S})$ and $\textsc{Diff}\mathrm{M}(\mathbb{A},\mathbb{B},\mathcal{S})$ is by decomposing the following graph.

The *family-free relational graph* $\mathit{FFR}(\mathbb{A},\mathbb{B},\mathcal{S})$, shown in Figure 1 (bottom), represents all possible weighted distances corresponding to all candidate ortholog-sets in $\mathcal{S}(\mathbb{A},\mathbb{B})$ [19]. Given a gene m, denote the extremities of m by m^h (*head*) and m^t (*tail*). The graph $\mathit{FFR}(\mathbb{A},\mathbb{B},\mathcal{S})$ has a set $V(\mathbb{A})$ with a vertex for each of the two extremities of each gene of genome $\mathbb{A}$ and a set $V(\mathbb{B})$ with a vertex for each of the two extremities of each gene of genome $\mathbb{B}$.

The set of edges is partitioned into several subsets:

- Sets $E^{\mathbb{A}}_{\mathrm{adj}}$ and $E^{\mathbb{B}}_{\mathrm{adj}}$ contain adjacency edges connecting adjacent extremities of genes in $\mathbb{A}$ and in $\mathbb{B}$.
- The set E_γ contains, for each edge $ab \in \mathcal{S}(\mathbb{A},\mathbb{B})$, an extremity edge connecting a^t to b^t, and an extremity edge connecting a^h to b^h. To both edges a^tb^t and a^hb^h, that are called *siblings*, we assign the same weight, which corresponds to the similarity of the edge ab in $\mathcal{S}(\mathbb{A},\mathbb{B})$: $w(a^tb^t) = w(a^hb^h) = \sigma(ab)$.
- Sets $E^{\mathbb{A}}_{\mathrm{id}}$ and $E^{\mathbb{B}}_{\mathrm{id}}$ contain indel edges connecting the two extremities of each gene in $\mathbb{A}$ and in $\mathbb{B}$. Each indel edge m^hm^t receives a weight $w(m^hm^t) = \max\{\sigma(mv)|mv \in \mathcal{S}(\mathbb{A},\mathbb{B})\}$, that is, it is the maximum similarity among the edges incident to the gene m in $\mathcal{S}(\mathbb{A},\mathbb{B})$.

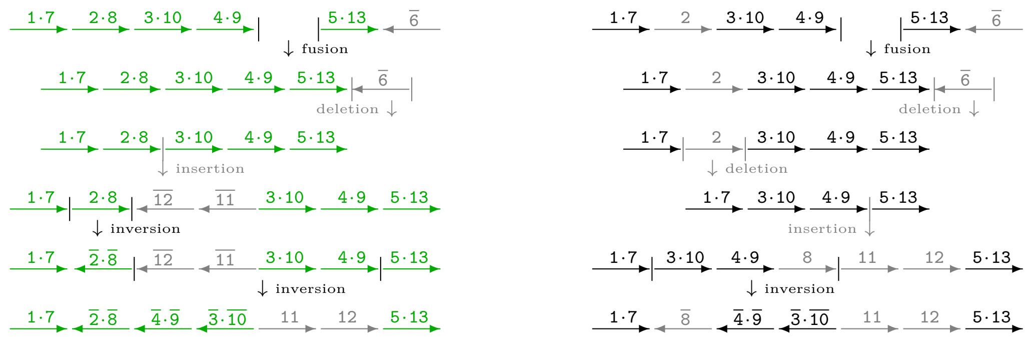

black ⊂ green (black lines next to green lines)
blue ⊂ black (blue dots on green/black lines)

Ranking of the represented ortholog-sets based on their corresponding distances

M	$\lvert M\rvert$	$w(M)$	$w(\widetilde{M})$	$d_{\text{DCJ}}^{\text{ID}}$	$wd_{\text{DCJ}}^{\text{ID}}$
`black`	4	3.2	1.8	5 $*$	7.6
`green`	5	3.5	1.2	5 $*$	7.7
`blue`	3	2.6	3.0	5	8.4
`magenta`	4	1.7	2.8	6	11.1

$*$ Rearrangement scenarios are given below, with genes belonging to $\widetilde{M}$ as well as indel operations represented in gray

`green` ortholog-set ($d_{\text{DCJ}}^{\text{ID}} = 5$): `black` ortholog-set ($d_{\text{DCJ}}^{\text{ID}} = 5$):



$FFR(\mathbb{A}, \mathbb{B}, \mathcal{S})$ and decomposition corresponding to the `black` ortholog-set of $\mathcal{S}(\mathbb{A}, \mathbb{B}, \mathcal{S})$

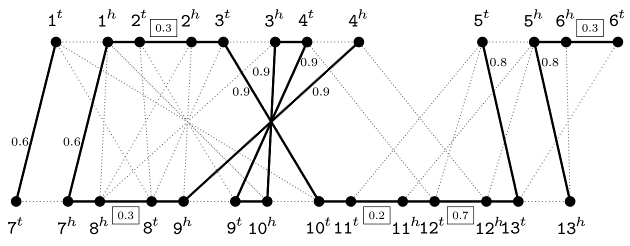


Figure 1 On the top part is displayed the gene similarity graph $\mathcal{S}(\mathbb{A}, \mathbb{B})$ of genomes $\mathbb{A} = \{[1\,2\,3\,4]\ [5\,\overline{6}]\}$ and $\mathbb{B} = \{[7\,\overline{8}\,\overline{9}\,\overline{10}\,11\,12\,13]\}$ and next to it a table with the ranking of four distinct ortholog-sets. On the middle the rearrangement scenarios induced by two of these ortholog-sets are shown. On the bottom part the family-free relational graph $FFR(\mathbb{A}, \mathbb{B}, \mathcal{S})$ is illustrated, highlighting the edges of the decomposition corresponding to the (`black`) ortholog-set $M = \{\{1,7\}, \{3,10\}, \{4,9\}, \{5,13\}\}$. (This decomposition has two $\mathbb{AB}$-paths, one $\mathbb{AA}$-path and one cycle.) All extremity and indel edges in $FFR(\mathbb{A}, \mathbb{B}, \mathcal{S})$ are weighted according to $\mathcal{S}(\mathbb{A}, \mathbb{B})$ but the weights of edges not derived from M or $\widetilde{M}$ are omitted.

2.3 Consistent decompositions of the family-free relational graph

A *decomposition* of $FFR(\mathbb{A}, \mathbb{B}, \mathcal{S})$ is a collection of vertex-disjoint *components*, that can be cycles and/or paths, covering all vertices of $FFR(\mathbb{A}, \mathbb{B}, \mathcal{S})$. We only consider *consistent* decompositions that correspond to ortholog-sets of $\mathcal{S}(\mathbb{A}, \mathbb{B})$. A set $S \subseteq E_\gamma$ is a *sibling-set* if it is exclusively composed of pairs of siblings and does not contain any pair of incident edges. Thus, a sibling-set S of $FFR(\mathbb{A}, \mathbb{B}, \mathcal{S})$ corresponds to an ortholog-set $M(S)$ of $\mathcal{S}(\mathbb{A}, \mathbb{B})$.

The set of edges $D[S]$ *induced* by a sibling-set S is said to be a *consistent decomposition* of $FFR(\mathbb{A}, \mathbb{B}, \mathcal{S})$ and can be obtained as follows. In the beginning, $D[S]$ is the union of S with the sets of adjacency edges $E_{\text{adj}}^{\mathbb{A}}$ and $E_{\text{adj}}^{\mathbb{B}}$. We then need to determine the *complement* of the sibling-set S, denoted by $\widetilde{S}$, that is composed of the indel-edges of $FFR(\mathbb{A}, \mathbb{B}, \mathcal{S})$ that must be added to $D[S]$: for each indel edge e, if its two endpoints have degree one or zero in $D[S]$, then e is added to both $\widetilde{S}$ and $D[S]$. (Note that $\widetilde{S}$ is equal to the complement of $M(S)$, while $|S| = 2|M(S)|$ and $w(S) = 2w(M(S))$.) The consistent decomposition $D[S]$ covers all vertices of $FFR(\mathbb{A}, \mathbb{B}, \mathcal{S})$ and is composed of cycles and paths. The paths connect the ends of linear chromosomes in both genomes and can be of three types: either $\mathbb{AA}$-path, or $\mathbb{BB}$-path or $\mathbb{AB}$-path.

The structure of $D[S]$ has all necessary information for computing $\text{wd}_{\text{DCJ}}^{\text{ID}}(\mathbb{A}, \mathbb{B}, \mathcal{S}, M(S))$, therefore we can say that $\text{wd}_{\text{DCJ}}^{\text{ID}}(\mathbb{A}, \mathbb{B}, \mathcal{S}, M(S)) = \text{wd}_{\text{DCJ}}^{\text{ID}}(D[S])$ [19] and modify our optimization problem to $\text{Diff}(\mathbb{A}, \mathbb{B}, \mathcal{S}) = \min_{S \in \mathfrak{S}}\{\text{wd}_{\text{DCJ}}^{\text{ID}}(D[S])\}$, where $\mathfrak{S}$ is the set of all possible sibling-sets in $FFR(\mathbb{A}, \mathbb{B}, \mathcal{S})$. Assuming that a decomposition $D[S_\star]$ gives the optimal solution for $\text{Diff}(\mathbb{A}, \mathbb{B}, \mathcal{S})$, then $\text{DiffM}(\mathbb{A}, \mathbb{B}, \mathcal{S}) = M(S_\star)$.

3 Capping

The end of a linear chromosome is called *telomere*. The telomeres are also the ends of the paths of any consistent decomposition. Therefore, if $\kappa(\mathbb{A})$ is the number of linear chromosomes in $\mathbb{A}$ and $\kappa(\mathbb{B})$ is the number of linear chromosomes in $\mathbb{B}$ the number of paths in any decomposition is $\kappa(\mathbb{A}) + \kappa(\mathbb{B})$. Our ILP is able to capture all necessary properties from the cycles of a decomposition, but cannot handle paths. A way to overcome this problem is by linking all paths of any decomposition with a known technique called *capping* [13].

3.1 Capping a consistent decomposition

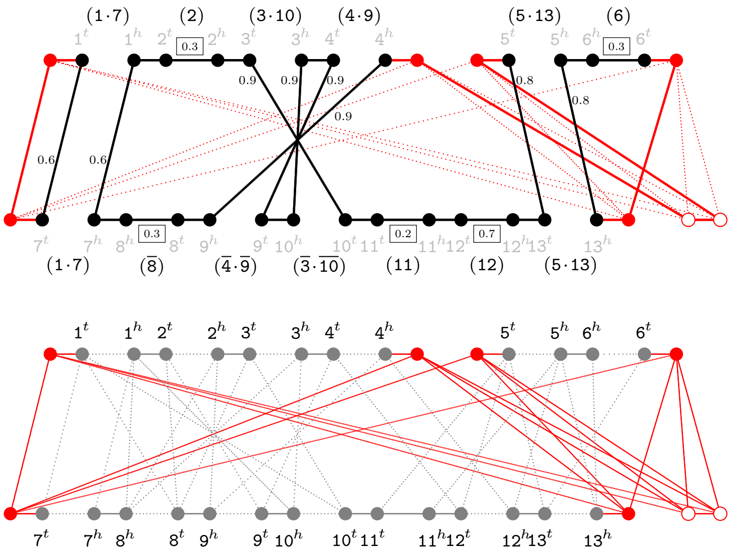
The idea of the capping is to split the telomeres into disjoint pairs and then to connect the two elements of each pair, so that all paths are linked into cycles. The only restriction is that a pair cannot contain telomeres from the same genome, therefore, if the numbers of telomeres in the two genomes are different, some *dummy* telomeres need to be created, as we describe in the following.

Suppose that $D[S]$ is any consistent decomposition of $FFR(\mathbb{A}, \mathbb{B}, \mathcal{S})$. For each telomere (vertex) v, add to $D[S]$ a *cap vertex* θ_v and connect v to θ_v by an adjacency edge. Now let $\theta(\mathbb{A})$ (respectively $\theta(\mathbb{B})$) be the set of all cap vertices in $\mathbb{A}$ (respectively in $\mathbb{B}$). Note that, since each linear chromosome has two ends, the cardinalities of these sets must be even. Moreover, if $|\theta(\mathbb{A})| \neq |\theta(\mathbb{B})|$, the cardinalities of these sets must be equalized. Let $p_* = \max\{\kappa(\mathbb{A}), \kappa(\mathbb{B})\}$ and $a_* = |\kappa(\mathbb{A}) - \kappa(\mathbb{B})|$. For equalizing the cardinalities with the minimum number of extra vertices, we need to add $2a_*$ extra cap vertices to the set with smaller cardinality. These extra cap vertices must be split into pairs (arbitrarily chosen) so that the vertices of each pair are connected by a *dummy* adjacency edge in $D[S]$. Denote by $\hat{\theta}(\mathbb{A})$ and $\hat{\theta}(\mathbb{B})$ the sets with equalized cardinalities and let P be a *capping-set*, which is a perfect matching between them: for $\gamma \in \hat{\theta}(\mathbb{A})$ and $\gamma' \in \hat{\theta}(\mathbb{B})$, if $\gamma\gamma' \in P$, then γ and γ' are connected by a *cap edge*. Let $\theta(D[S], P)$ be a *capped decomposition* of $D[S]$ with capping-set P. It is easy to see that $\theta(D[S], P)$ is composed of cycles only.

DCJ-indel optimal capping

So far we explained how to guarantee that all paths in any decomposition are linked into cycles. Note, however, that there are $(2p_*)!$ ways of completely matching the vertices of sets $\hat{\theta}(\mathbb{A})$ and $\hat{\theta}(\mathbb{B})$. For a given decomposition $D[S]$, any of these possibilities, say capping-set

P, would produce a capped decomposition $\theta(D[S], P)$, and capping the same $D[S]$ with distinct capping-sets may produce distinct weighted costs. Let $P_\star$ be an optimal capping-set for $D[S]$, that is, the capped decomposition $\theta(D[S], P_\star)$ has the minimum weighted cost among all capped decompositions of $D[S]$. (There can be several co-optimal capping-sets for the same decomposition $D[S]$ and each optimal capping-set links up to 4 cycles of $D[S]$ into a single cycle [5].) It has been shown that $\mathrm{wd}^{\mathrm{ID}}_{\mathrm{DCJ}}(\theta(D[S], P_\star)) = \mathrm{wd}^{\mathrm{ID}}_{\mathrm{DCJ}}(D[S])$, therefore any consistent decomposition of $D[S]$ with an optimal capping-set is DCJ-indel optimal and preserves the weighted cost of $D[S]$, reporting both $\mathrm{d}^{\mathrm{ID}}_{\mathrm{DCJ}}(\mathbb{A}, \mathbb{B}, M(S))$ and $\mathrm{wd}^{\mathrm{ID}}_{\mathrm{DCJ}}(\mathbb{A}, \mathbb{B}, \mathcal{S}, M(S))$ [19]. Figure 2 (top) highlights an optimal capping of a consistent decomposition.



■ **Figure 2** On the top part we show the capping of the decomposition corresponding to the (`black`) ortholog-set $M = \{\{1, 7\}, \{3, 10\}, \{4, 9\}, \{5, 13\}\}$ from the gene similarity graph $\mathcal{S}(\mathbb{A}, \mathbb{B})$ of Figure 1 (bottom). Each red vertex is a cap vertex. Each filled (red) vertex is connected to a telomere (chromosome/path ends). The unfilled vertices represent the extra (equalizing) vertices connected by a dummy adjacency. The capping is a perfect matching of the complete bipartite graph of the cap vertices. The optimal capping for this decomposition is highlighted. It closes each of its paths into a separate cycle. (In general, an optimal capping of a decomposition may link up to 4 paths into a single cycle [5]). On the bottom part is displayed the complete family-free graph $FFR(\mathbb{A}, \mathbb{B}, \mathcal{S})$ optimally capped. Cap edges are unweighted. Weights of extremity and indel edges are omitted.

3.2 Optimally capped family-free relational graph

All consistent decompositions share the same telomeres, therefore a set of capping-sets for one decomposition is also a set of capping-sets of any other decomposition. If we then simply add all possible capping-sets to the family-free relational graph, which implies adding a complete bipartite graph with partite sets $\hat{\theta}(\mathbb{A})$ and $\hat{\theta}(\mathbb{B})$, we guarantee that an optimal solution can be found. Let the so-called optimal capping (represented in Figure 2 (bottom)) of $FFR(\mathbb{A}, \mathbb{B}, \mathcal{S})$ with the minimum number of extra elements be denoted by $\theta_\star(FFR(\mathbb{A}, \mathbb{B}, \mathcal{S}))$ and be defined as follows:

1. Add the set of *cap vertices* $\hat{\theta}(\mathbb{A}) = \theta_{\mathbb{A}}^1, \theta_{\mathbb{A}}^2, \ldots, \theta_{\mathbb{A}}^{2p_*}$ and connect each telomere of genome $\mathbb{A}$ to one of these cap vertices by an adjacency edge added to $E_{\text{adj}}^{\mathbb{A}}$.
2. Similarly, add the set of cap vertices $\hat{\theta}(\mathbb{B}) = \theta_{\mathbb{B}}^1, \theta_{\mathbb{B}}^2, \ldots, \theta_{\mathbb{B}}^{2p_*}$ and connect each telomere of genome $\mathbb{B}$ to one of these cap vertices by an adjacency edge added to $E_{\text{adj}}^{\mathbb{B}}$.
3. Add (arbitrarily chosen) $p_* - \kappa(\mathbb{A})$ dummy adjacency edges to $E_{\text{adj}}^{\mathbb{A}}$ and $p_* - \kappa(\mathbb{B})$ dummy adjacency edges to $E_{\text{adj}}^{\mathbb{B}}$. (Note that only one of the two genomes may have dummy adjacencies.)
4. Connect all cap vertices in $\hat{\theta}(\mathbb{A})$ to all cap vertices in $\hat{\theta}(\mathbb{B})$ with *cap edges*. The set of all cap edges is denoted by E_θ.

Since all $2p_*$ cap vertices in $\mathbb{A}$ are connected to all $2p_*$ cap vertices in $\mathbb{B}$ and any perfect matching of these edges is a valid capping, the search space of our optimization problem is multiplied by $(2p_*)!$. Denote by $\mathfrak{P}$ the set of all possible capping-sets (perfect matchings) between the vertices from $\hat{\theta}(\mathbb{A})$ and $\hat{\theta}(\mathbb{B})$. The optimization problem over $\theta_\star(\mathit{FFR}(\mathbb{A}, \mathbb{B}, \mathcal{S}))$ can be rewritten as $\textsc{Diff}(\mathbb{A}, \mathbb{B}, \mathcal{S}) = \min\limits_{S \in \mathfrak{S}, P \in \mathfrak{P}} \{\text{wd}_{\text{DCJ}}^{\text{ID}}(\theta(D[S], P)\}$. Assuming that an optimal capping of a decomposition $D[S_\star]$ gives the optimal solution for $\textsc{Diff}(\mathbb{A}, \mathbb{B}, \mathcal{S})$, the optimal ortholog-set is $\textsc{DiffM}(\mathbb{A}, \mathbb{B}, \mathcal{S}) = M(S_\star)$. Both problems $\textsc{Diff}$ and $\textsc{DiffM}$ can be solved with the ILP formulation FF-DCJ-$\textsc{Indel}$ [19], which can be found in Appendix A.

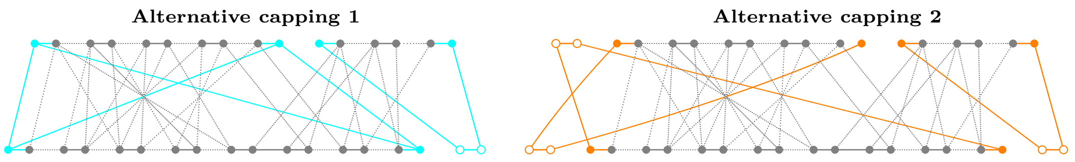
3.3 Integration of pairwise optimal ortholog-sets into gene families

In our previous work [18], the ILP FF-DCJ-$\textsc{Indel}$ solving $\textsc{DiffM}$ (with optimal capping) was integrated in a tool called $\textsc{Diff}$MGC for inferring gene families across several species. The pipeline, illustrated in Figure 6 of Appendix B, can be summarized as follows: given a set of n genomes, gene similarities and ortholog-sets are computed for all pairwise comparisons and simply integrated into an n-partite graph. The connected components of this graph are the inferred gene families.

4 Heuristic capping

Conceptually our approach can handle partially assembled genomes distributed into several contigs/scaffolds: each of these is a linear segment and could simply be treated as the same object that we so far called chromosome. However, as already explained, the optimal capping multiplies the search space of $\mathit{FFR}(\mathbb{A}, \mathbb{B})$ by $(2p_*)!$ where p_* is the maximum between the number of linear segments in genomes $\mathbb{A}$ and $\mathbb{B}$. This effect makes it unfeasible to analyze genomes with a large number of segments with our ILP over an optimally capped family-free relational graph.

One way of overcoming this issue is by adopting a lighter capping, for example by removing some edges from the complete bipartite graph of $\hat{\theta}(\mathbb{A})$ and $\hat{\theta}(\mathbb{B})$, and/or by partitioning these sets into subsets that are capped independently. In any case it is important to guarantee that a capping is *valid*, that is, it allows to find a capping-set (a perfect matching of the cap vertices). A valid lighter capping may not include the optimal capping-sets, and therefore may not preserve the computed weighted costs. Note, however, that even if the weighted costs are not preserved, the ranking of the ortholog-sets/sibling-sets may not be affected. In Figure 3 we show examples of arbitrary lighter valid cappings and their effects on the sibling-set ranking.



Ortholog-sets and their corresponding costs with the two alternative lighter cappings above

				Optimal		Alternative 1		Alternative 2	
M	$\lvert M\rvert$	$w(M)$	$w(\widetilde{M})$	$\mathrm{d}^{\mathrm{ID}}_{\mathrm{DCJ}}$	$\mathrm{wd}^{\mathrm{ID}}_{\mathrm{DCJ}}$	$\mathrm{d}^{\mathrm{ID}}_{\mathrm{DCJ}}$	$\mathrm{wd}^{\mathrm{ID}}_{\mathrm{DCJ}}$	$\mathrm{d}^{\mathrm{ID}}_{\mathrm{DCJ}}$	$\mathrm{wd}^{\mathrm{ID}}_{\mathrm{DCJ}}$
`black`	4	3.2	1.8	5	7.6	6	8.6	8	10.6
`green`	5	3.5	1.2	5	7.7	6	8.7	8	10.7
`blue`	3	2.6	3.0	5	8.4	6	9.4	7	10.4
`magenta`	4	1.7	2.8	6	11.1	6	11.1	7	12.1

Figure 3 Examples of two arbitrary lighter valid cappings of the family-free relational diagram from Figure 1 (bottom) and their effects on the ranking of ortholog-sets/sibling-sets represented in Figure 1 (top). Both cappings affect the computed distances, but, while the capping shown in the left (cyan) preserves the optimal ranking, the one shown in the right (orange) does not.

4.1 Perfect contig intersection graph with thresholds τ and ϵ

Our goal is therefore to develop a lighter heuristic capping that may potentially preserve the original (optimal) ranking of the best ortholog-sets/sibling-sets. We achieve this by connecting cap vertices only between the telomeres of the linear segments (contigs or chromosomes) that (potentially) share most of their genomic contents. This is because those telomeres have a higher chance of being in the same paths of the best consistent decompositions of the family-free relational graph.

Given two contigs α and β belonging to genomes $\mathbb{A}$ and $\mathbb{B}$, respectively, their *shared genomic content* $\lambda(\alpha, \beta)$ is the number of edges in $\mathcal{S}$ between genes of α and β. Formally, for $\alpha \subseteq \mathbb{A}$, $\beta \subseteq \mathbb{B}$, we have $\lambda(\alpha, \beta) = |\{g_1 g_2 \in \mathcal{S} \mid g_1 \in \alpha, g_2 \in \beta \text{ and } w(g_1 g_2) > 0\}|$. Now let $\mathcal{C}(\mathbb{A}, \mathbb{B}) = (V_\mathbb{A}, V_\mathbb{B}, E)$ be the bipartite *contig intersection graph* where the vertex sets are $V_\mathbb{A} = \{\alpha : \alpha \text{ is a contig in } \mathbb{A}\}$ and $V_\mathbb{B} = \{\beta : \beta \text{ is a contig in } \mathbb{B}\}$. Initially the set of edges is $E = \{\alpha\beta \mid \alpha \in V_\mathbb{A}, \beta \in V_\mathbb{B} \text{ and } \lambda(\alpha, \beta) > 0\}$. Each edge $\alpha\beta$ is weighted such that $w(\alpha\beta) = \lambda(\alpha\beta)$. Then, given a positive integer τ, we remove some edges from E by applying a filtering procedure that simply iterates over $V_\mathbb{A} \cup V_\mathbb{B}$, keeping in E only the τ edges of highest weights for each vertex. Then, the remaining edges are again filtered out to remove weak relations between contigs, according to another threshold given by a rational value $\epsilon \in [0, 1]$: for each vertex v and the edge e of highest weight incident to v, edges uv of weights below $\epsilon w(e)$ are removed.

Capping attempt induced by the contig intersection graph

The contig intersection graph will now *induce* our capping procedure. The idea is to allow cap connections only between the ends of contigs that are connected in $\mathcal{C}$. Therefore, the contigs that are in the same connected component of $\mathcal{C}$ will be capped together, independently from the contigs that are in other connected components. In other words, the connected components of $\mathcal{C}$ will impose a partitioning of the capping procedure.

Let us then assume that $\mathcal{C}$ has a single connected component. Note that a capping induced by $\mathcal{C}$ can only be valid if its partite sets $V_\mathbb{A}$ and $V_\mathbb{B}$ are of the same size. This necessary condition also applies and is sufficient for the optimal capping, but here it is not sufficient: even when $V_\mathbb{A}$ and $V_\mathbb{B}$ are of the same size, since not all connections between the ends of the contigs in $V_\mathbb{A}$ and in $V_\mathbb{B}$ are present, the induced capping could be invalid (Figure 4 (a)).

Here the necessary and sufficient condition for a valid capping is the existence of a perfect matching in $\mathcal{C}(\mathbb{A},\mathbb{B})$, as stated in Lemma 2, whose proof relies on a theorem closely related to perfect matchings and demonstrated by Hall [12] in 1935. Denote by $\mathcal{N}(S)$ the neighborhood of a vertex set S, that is, the set of all vertices adjacent to some vertex of S.

▶ **Theorem 1** (Hall's marriage theorem). *Let $G = (U, V, E)$ be a bipartite graph. There exists a matching in G that covers the vertex set V if and only if for each subset $S \subseteq V$, $|S| \leq |\mathcal{N}(S)|$.*

Note that a perfect matching exists in $\mathcal{C}$ if and only if the condition of Theorem 1 holds for both $V_{\mathbb{A}}$ and $V_{\mathbb{B}}$. We can now establish the relation between perfect matchings in $\mathcal{C}$ and the validity of the capping it induces in the family-free relational graph.

▶ **Lemma 2.** *A perfect matching exists in $\mathcal{C}(\mathbb{A},\mathbb{B})$ if and only if the capping of $FFR(\mathbb{A},\mathbb{B},\mathcal{S})$ induced by $\mathcal{C}$ is valid.*

Proof. If a perfect matching M exists in $\mathcal{C}$, for each edge $\alpha\beta$ in M, in the induced capping of $FFR(\mathbb{A},\mathbb{B},\mathcal{S})$ the pair of cap vertices connected to the ends of α can be connected in any of the two distinct ways to the pair of cap vertices connected to the ends of β, resulting in a capping-set.

The converse is shown by contraposition. Suppose that a maximum cardinality matching M in $\mathcal{C}$ is not a perfect matching. Therefore, by Hall's marriage theorem, there exists some S in $\mathcal{C}$ such that $|\mathcal{N}(S)| < |S|$. Let S' be the set of cap vertices in the capping induced by $\mathcal{C}$ for all contigs in S. Since the connection of these cap vertices follows $\mathcal{C}$ and each contig has two cap vertices, it is clear that $|S'| = 2|S|$ and $|\mathcal{N}(S')| = 2|\mathcal{N}(S)|$, hence, $|\mathcal{N}(S')| < |S'|$. By the pigeonhole principle, at least 2 cap vertices (because $|\mathcal{N}(S')|$ and $|S'|$ are even numbers) will not be incident to any cap edge, therefore no capping-set exists. ◀

Building the perfect contig intersection graph

We will now describe a procedure for transforming the contig intersection graph $\mathcal{C}(\mathbb{A},\mathbb{B})$ into a *perfect contig intersection graph* $\widehat{\mathcal{C}}(\mathbb{A},\mathbb{B})$ that has at least one perfect matching, as shown in Algorithm 1. In the completion loop, dummy contigs are iteratively created until a perfect matching is possible. If a maximum cardinality matching M is found but is not a perfect matching, a Hall violator set S can be found as follows. Let v be a vertex unsaturated by M, then $S = \{v\} \cup \{u \mid u \text{ is reachable from } v \text{ by an } M\text{-alternating path}\}$. Finally, $|S| - |\mathcal{N}(S)|$ dummy contigs are created and connected to each contig in S.

An edge in $\widehat{\mathcal{C}}(\mathbb{A},\mathbb{B})$ is *matchable* if it is part of at least one perfect matching and *non-matchable* otherwise. Once the completion loop is finished, $\widehat{\mathcal{C}}$ admits at least one perfect matching and its matchable edges can be identified efficiently [23]. The last step of our algorithm is then removing from $\widehat{\mathcal{C}}$ all non-matchable edges. An example of the construction of a perfect contig intersection graph is illustrated in Figure 4 (b).

Search space compared to optimal capping

If the threshold τ is similar to the numbers $\kappa(\mathbb{A})$ and $\kappa(\mathbb{B})$ of contigs in each genome, and in the unlike situation where all contigs from one genome are connected to all contigs from the other genome in $\mathcal{C}$, the heuristic capping induced by $\widehat{\mathcal{C}}$ may be as large as the optimal capping.

The threshold τ is thought to be smaller than $\kappa(\mathbb{A})$ and $\kappa(\mathbb{B})$, effectively reducing the number of capping-sets. We could not yet estimate this reduction as a function of τ, though. As our experimental results with real genomes show (details below, in Section 5), with a small τ the heuristic capping leans to a considerably smaller number of capping-sets in practice.

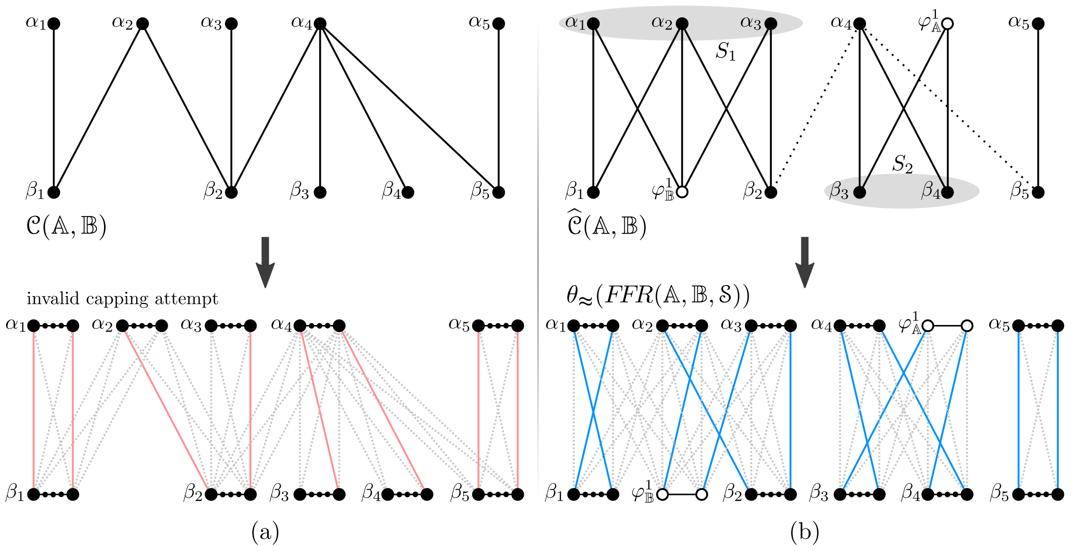


Figure 4 (a) Example of a contig intersection graph $\mathcal{C}(\mathbb{A}, \mathbb{B})$. The genomes $\mathbb{A}$ and $\mathbb{B}$ have contigs $\alpha_{1..5}$ and $\beta_{1..5}$, respectively. The capping of $FFR(\mathbb{A}, \mathbb{B}, \mathcal{S})$ induced by $\mathcal{C}$ is invalid. (b) Transformation of $\mathcal{C}$ into a perfect contig intersection graph $\widehat{\mathcal{C}}(\mathbb{A}, \mathbb{B})$: Vertex sets S_1 and S_2 represent Hall violators (among other possibilities) that demand the creation of dummy contigs $\varphi^1_{\mathbb{B}}$ and $\varphi^1_{\mathbb{A}}$, respectively. Dotted edges represent those that are non-matchable and must be removed from $\widehat{\mathcal{C}}$ after the completion is finished. Notice that the component with vertices $\alpha_1, \alpha_2, \alpha_3, \beta_1, \varphi^1_{\mathbb{B}}$ and β_2 is not a complete bipartite subgraph. (In both (a) and (b), for the capped *FFR* only cap vertices, cap edges and dummy adjacencies are represented explicitly, while vertices of gene extremities between cap vertices are represented by a line with small dots. In addition, colored solid edges represent a maximum cardinality matching between cap vertices, while the cap edges not in the matching are dashed grey.)

4.2 Heuristically capped family-free relational graph

The bipartite perfect contig intersection graph $\widehat{\mathcal{C}}(\mathbb{A}, \mathbb{B})$ has edge set $\widehat{E}$ and partite sets $\widehat{V}_{\mathbb{A}} = V_{\mathbb{A}} \cup V^{\varphi}_{\mathbb{A}}$ and $\widehat{V}_{\mathbb{B}} = V_{\mathbb{B}} \cup V^{\varphi}_{\mathbb{B}}$, where $V_{\mathbb{A}}$ and $V_{\mathbb{B}}$ are the sets of contigs and $V^{\varphi}_{\mathbb{A}}$ and $V^{\varphi}_{\mathbb{B}}$ the sets of dummy contigs. Recall that the sets $\widehat{V}_{\mathbb{A}}$ and $\widehat{V}_{\mathbb{B}}$ have the same cardinality, which here we denote by $p_{\approx}$. The heuristic capping $\theta_{\approx}$ of the family-free relational graph $FFR(\mathbb{A}, \mathbb{B}, \mathcal{S})$ induced by $\widehat{\mathcal{C}}(\mathbb{A}, \mathbb{B})$ is shown in Figure 4 (b) and described as follows:

1. Add the set of cap vertices $\hat{\theta}(\mathbb{A}) = \theta^1_{\mathbb{A}}, \theta^2_{\mathbb{A}}, \ldots, \theta^{2p_{\approx}}_{\mathbb{A}}$. For $i = 1 \ldots |V_{\mathbb{A}}|$, associate each contig $\alpha_i \in V_{\mathbb{A}}$ to cap vertices $\theta^{2i-1}_{\mathbb{A}}$ and $\theta^{2i}_{\mathbb{A}}$ and connect with adjacency edges one telomere of α_i to $\theta^{2i-1}_{\mathbb{A}}$ and the other to $\theta^{2i}_{\mathbb{A}}$. Note that $2|V^{\varphi}_{\mathbb{A}}|$ cap vertices remain disconnected.
2. Similarly, add cap vertices $\hat{\theta}(\mathbb{B}) = \theta^1_{\mathbb{B}}, \theta^2_{\mathbb{B}}, \ldots, \theta^{2p_{\approx}}_{\mathbb{B}}$. For $j = 1 \ldots |V_{\mathbb{B}}|$, associate each contig $\beta_j \in V_{\mathbb{B}}$ to cap vertices $\theta^{2j-1}_{\mathbb{B}}$ and $\theta^{2j}_{\mathbb{B}}$ and connect with adjacency edges one telomere of β_i to $\theta^{2j-1}_{\mathbb{B}}$ and the other to $\theta^{2j}_{\mathbb{B}}$. Again, $2|V^{\varphi}_{\mathbb{B}}|$ cap vertices remain disconnected.
3. For $i_\circ = 1 \ldots |V^{\varphi}_{\mathbb{A}}|$ and $i = |V_{\mathbb{A}}| + i_\circ$, connect the pair of cap vertices $\theta^{2i-1}_{\mathbb{A}}$ and $\theta^{2i}_{\mathbb{A}}$ by a dummy adjacency edge, associating this pair to the dummy contig $\varphi^{i_\circ}_{\mathbb{A}} \in V^{\varphi}_{\mathbb{A}}$. Similarly, for $j_\circ = 1 \ldots |V^{\varphi}_{\mathbb{B}}|$ and $j = |V_{\mathbb{B}}| + j_\circ$, connect the pair of cap vertices $\theta^{2j-1}_{\mathbb{B}}$ and $\theta^{2j}_{\mathbb{B}}$ by a dummy adjacency edge, associating this pair to the dummy contig $\varphi^{j_\circ}_{\mathbb{B}} \in V^{\varphi}_{\mathbb{B}}$.
4. For each edge $ab \in \widehat{E}$, let $a \in \widehat{V}_{\mathbb{A}}$ and $b \in \widehat{V}_{\mathbb{B}}$ be associated, respectively, to the cap vertices $\theta^{2i-1}_{\mathbb{A}}, \theta^{2i}_{\mathbb{A}} \in \hat{\theta}(\mathbb{A})$ and $\theta^{2j-1}_{\mathbb{B}}, \theta^{2j}_{\mathbb{B}} \in \hat{\theta}(\mathbb{B})$. Connect the four "crossing" pairs of cap vertices $\{\theta^{2i-1}_{\mathbb{A}}, \theta^{2j-1}_{\mathbb{B}}\}$, $\{\theta^{2i}_{\mathbb{A}}, \theta^{2j-1}_{\mathbb{B}}\}$, $\{\theta^{2i-1}_{\mathbb{A}}, \theta^{2j}_{\mathbb{B}}\}$ and $\{\theta^{2i}_{\mathbb{A}}, \theta^{2j}_{\mathbb{B}}\}$ with cap edges. The set of all cap edges is denoted by E_θ.

Algorithm 1 Creates a perfect contig intersection graph from a contig intersection graph.

```
Input: A contig intersection graph $\mathcal{C}(\mathbb{A},\mathbb{B}) = (V_{\mathbb{A}}, V_{\mathbb{B}}, E)$
Output: A perfect contig intersection graph $\widehat{\mathcal{C}}(\mathbb{A},\mathbb{B}) = (\widehat{V}_{\mathbb{A}}, \widehat{V}_{\mathbb{B}}, \widehat{E})$
1: $\widehat{V}_{\mathbb{A}} \leftarrow V_{\mathbb{A}}$ ,  $\widehat{V}_{\mathbb{B}} \leftarrow V_{\mathbb{B}}$ ,  $\widehat{E} \leftarrow E$
2: while a maximum cardinality matching $M$ in $\widehat{E}$ is not a perfect matching do
3:     $S \leftarrow$ a Hall violator set derived from $M$
4:     $\Phi \leftarrow$ dummy contigs $\{\varphi^1, \ldots, \varphi^{|S|-|\mathcal{N}(S)|}\}$
5:     if $S \subseteq \widehat{V}_{\mathbb{A}}$ then $\widehat{V}_{\mathbb{B}} \leftarrow \widehat{V}_{\mathbb{B}} \cup \Phi$ else $\widehat{V}_{\mathbb{A}} \leftarrow \widehat{V}_{\mathbb{A}} \cup \Phi$
6:     $\widehat{E} \leftarrow \widehat{E} \cup (S \times \Phi)$
7: remove from $\widehat{E}$ all non-matchable edges
8: return $\widehat{\mathcal{C}} = (\widehat{V}_{\mathbb{A}}, \widehat{V}_{\mathbb{B}}, \widehat{E})$
```

Denote by $\mathfrak{P}_{\approx}$ the set of all possible capping-sets (perfect matchings) between the vertices of $\hat{\theta}(\mathbb{A})$ and $\hat{\theta}(\mathbb{B})$. The optimization problem over $\theta_{\approx}(\mathit{FFR}(\mathbb{A},\mathbb{B},\mathcal{S}))$ is defined as

$$\textsc{Diff}\overset{\approx}{\text{H}}(\mathbb{A},\mathbb{B},\mathcal{S}) = \min_{S\in\mathfrak{S}, P\in\mathfrak{P}_{\approx}} \{\mathrm{wd}^{\mathrm{ID}}_{\mathrm{DCJ}}(\theta(D[S], P)\}\,.$$

Assuming that a heuristic capping-set of a decomposition $D[S_{\approx}]$ gives the optimal solution for $\textsc{Diff}\overset{\approx}{\text{H}}(\mathbb{A},\mathbb{B},\mathcal{S})$, the best heuristic ortholog-set is $\textsc{DiffM}\overset{\approx}{\text{H}}(\mathbb{A},\mathbb{B},\mathcal{S}) = M(S_{\approx})$. Both problems $\textsc{Diff}\overset{\approx}{\text{H}}$ and $\textsc{DiffM}\overset{\approx}{\text{H}}$ can also be solved with the ILP FF-DCJ-Indel, shown in Appendix A.

4.3 Integration of pairwise heuristic ortholog-sets into gene families

The ILP FF-DCJ-Indel solving $\textsc{DiffM}\overset{\approx}{\text{H}}$ (with heuristic capping) is the core of a new version of our tool, called $\textsc{DiffMGC}\overset{\approx}{\text{H}}$, for inferring gene families across several species, as illustrated in Figure 6 of Appendix B. Recall that each family is a connected component of the n-partite graph obtained by the simple integration of the computed pairwise ortholog-sets. An *ambiguous* family corresponds to a connected component of the n-partite graph that has more than one gene from the same genome. Otherwise we have a *resolved* family, which can be either *complete*, when it contains one gene per genome, or *incomplete* otherwise. The types of families are shown in Figure 7 of Appendix B.

5 Implementation and experiments

The pipeline DiffMGC (with optimal capping) was previously integrated into the ffgc workflow [11, 19], which includes the pre-computation of gene similarities, allowing therefore the automatic generation of families directly from the genome data. We implemented the new pipeline $\textsc{DiffMGC}\overset{\approx}{\text{H}}$ (with heuristic capping) as another extension of the same workflow. The implementation and its documentation can be downloaded from our GitLab server at `gitlab.ub.uni-bielefeld.de/gi/FFGC`.

5.1 Computational environment and parameters

We ran experiments in a 2.4GHz multi-core machine. Whenever possible, tasks ran using 8 cores. As an ILP solver, we used CPLEX. The default values of the parameters of DiffMGC and $\textsc{DiffMGC}\overset{\approx}{\text{H}}$ for the pre-computation of gene similarities (with the help of blast [2]) were kept, except for two of them. The first one is the minimum number of genomes for which each gene must share some similarity in, set to 1 – otherwise genes not similar to any other gene, which should be still considered in indels, will not appear in genomes. The second one is the stringency threshold, set to $t = 0.8$. Succinctly, the stringency filter [11, 14] prunes edges in $\mathcal{S}(A, B)$ that are adjacent to edges with considerably higher weights based on a threshold. Empirical evidence shows that thresholds higher than $t = 0.8$ may discard relevant gene relationships, while lower values simply increase the ILP running time with small variations in the results found by the solver.

For the capping heuristic, we set $\tau = 3$ (which in average corresponds to half of the number of chromosomes in a completely assembled *Drosophila* genome) and $\epsilon = 0.01$ (a conservative choice for only filtering out from $\widehat{\mathcal{C}}$ very weak edges). Since these (reasonable) choices produced very good results, we did not explore other possibilities, but in a future work we intend to do a systematic study testing the speed and quality performances of the heuristic capping for different values of τ and ϵ.

5.2 Analysis of *Drosophila* genomes

All genome assemblies used in experiments were fetched from NCBI. The FlyBase consortium sequenced, assembled and annotated the genomes of 12 *Drosophilas* with $\sim$ 12,000–16,000 protein-coding genes, however only 11 of those genomes are available on NCBI together with the complete annotation: *D. simulans*, *D. sechellia*, *D. melanogaster*, *D. yakuba*, *D. erecta*, *D. ananassae*, *D. persimilis*, *D. willistoni*, *D. mojavensis*, *D. virilis* and *D. grimshawi*.

Average numbers of cap edges and capping-sets in $\theta_\star$ and in $\theta_\approx$

The analyzed *Drosophila* genomes have 507 contigs on average, therefore each optimally capped family-free relational diagram has $1{,}014 \times 1{,}014 = 1{,}028{,}196$ cap edges and an unfeasible total of 1,014! capping-sets on average.

In contrast, considering the perfect contig intersection graphs for all pairwise *Drosophila* comparisons, 99.7% of the components in those graphs have only 1 contig in each part of the graph. In the remaining 0.3%, 80% have 7 or fewer contigs in each part, with the largest component having 76 contigs in each part. The perfect contig intersection graphs have an average of 1,419 edges. For that number of edges, each heuristically capped family-free relational diagram has 2,838 cap edges on average. As the exact number of perfect matchings in arbitrary graphs is not trivial to estimate, we computed an upper limit for the average number of distinct capping-sets: $\sim$ 305!.

Benchmark for our experiments

Reference families were obtained directly from FlyBase (`flybase.org`). Since the set of genes classified in FlyBase is slightly different from the set of genes present with their coding sequences in database files, we filtered out a small portion ($\sim$ 7%) of genes in FlyBase families so that only those present in the databases with their coding sequences were kept. Prior to any comparison of inferred families to FlyBase families, we also filtered out from the inferred families genes not present in FlyBase families.

We perform two distinct comparisons. First, based on a set of three completely assembled *Drosophila* genomes, we compare DiffMGC$\overset{\approx}{\text{H}}$ families (inferred with the optimal capping) to DiffMGC families (inferred with the heuristic capping) using FlyBase families as reference. Next, based on the complete set of 11 *Drosophilas* including partially assembled genomes, we compare DiffMGC$\overset{\approx}{\text{H}}$ families to the gene families inferred by other inference tools, again using FlyBase families as reference.

5.2.1 Comparing DiffMGC$\widetilde{\widetilde{\mathrm{H}}}$ to DiffMGC

We performed a first empirical evaluation on how the heuristic capping could impact running times and the quality of results. We compare families inferred by DiffMGC and DiffMGC$\widetilde{\widetilde{\mathrm{H}}}$, considering genomes of *D. melanogaster*, *D. simulans* and *D. yakuba*, with 7, 6 and 6 linear chromosomes (after filtering out unlocalized contigs), respectively. DiffMGC cannot deal with large contig numbers and those were the only species assembled by FlyBase at chromosome level available on NCBI.

For DiffMGC$\widetilde{\widetilde{\mathrm{H}}}$, the largest number of edges in $\widehat{\mathcal{C}}(\mathbb{A}, \mathbb{B})$ was for $A = D.\,melanogaster$ and $B = D.\,yakuba$. In this case, $\widehat{\mathcal{C}}$ has 10 edges distributed among 4 components with 2 vertices (including 1 dummy contig), and 1 component with 6 vertices. That corresponds to at most 11,520 capping-sets in $\theta_{\approx}$, while the three pairwise comparisons with the optimal $\theta_{\star}$ have between 12! and 14! capping-sets. Even so, the running times were very similar between the two approaches, varying from 15 to 30 seconds, probably due to the fact that these three species are phylogenetically closely related and very well assembled and annotated, allowing the solver to quickly identify the best ortholog-sets despite the much higher increase of the search space produced by $\theta_{\star}$.

Quality of inferred families

While 12,406 families were inferred using DiffMGC, 12,405 were inferred using DiffMGC$\widetilde{\widetilde{\mathrm{H}}}$, and 99.8% of those families are the same. Consequently, only slight variations were found when comparing those families to the FlyBase families – 11,542 and 11,544 families are identical to those of FlyBase for DiffMGC and DiffMGC$\widetilde{\widetilde{\mathrm{H}}}$, respectively.

5.2.2 Comparing DiffMGC$\widetilde{\widetilde{\mathrm{H}}}$ to other tools

For comparing the performance of DiffMGC$\widetilde{\widetilde{\mathrm{H}}}$ against other inference tools, we analyzed the complete dataset with 11 *Drosophila* genomes. Unlocalized contigs were not filtered out, resulting in genomes with 11 to 1,041 contigs.

Homologous families were also inferred by the following tools using the default parameters unless noted otherwise.

ProteinOrtho and Poff. ProteinOrtho [14] compares similarities of gene sequences and clusters them to find significant orthologous groups. To enhance the prediction accuracy, the Poff extension [15] can be used to take into account the relative order of genes as an additional feature for the discrimination of orthologs. We changed the parameter "minimum reciprocal similarity for additional hits" from the default 0.95 to 0.8 – experiments have shown that, as the "stringency threshold" parameter of DiffMGC, higher values might discard relevant gene relationships.

Oma. Based on sequence similarities and on phylogeny, Oma [1] is the underlying tool of the homonym online orthology browser. The standalone tool allows custom genomes to be compared to infer orthologous groups. We have also provided the phylogenetic tree of the 11 *Drosophilas* as input.

Quality of inferred families

Considering FlyBase families as reference, we compare the families inferred by Oma, ProteinOrtho, Poff and DiffMGC$\overset{\approx}{\text{H}}$ (Figure 5). All methods inferred more than 12,000 families. Since the number of families alone cannot hint the quality of results, we focus on the intersections with FlyBase families.

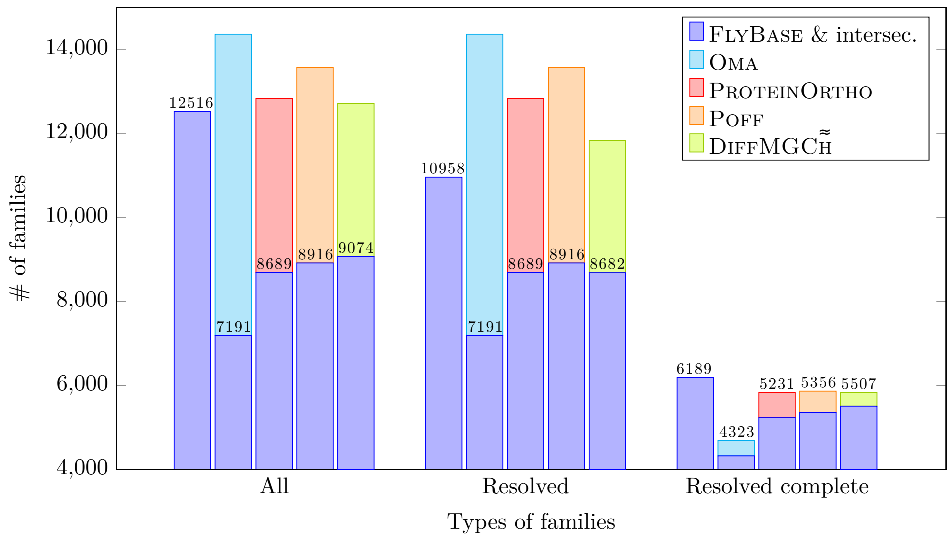


Figure 5 The numbers of all, resolved and resolved complete families in FlyBase, followed by the numbers of families inferred by Oma, ProteinOrtho, Poff and DiffMGC$\overset{\approx}{\text{H}}$, respectively. The lower part of each bar represents the intersection between the inferred sets and FlyBase. (For resolved complete families, the numbers of families in the intersections are shown on the top of bars.) More details on the distribution of families inferred by the four methods can be found in Appendix C.

All except DiffMGC$\overset{\approx}{\text{H}}$ produced only resolved families. For general and resolved families, 7,191 Oma families and FlyBase families are identical. ProteinOrtho, Poff and DiffMGC$\overset{\approx}{\text{H}}$ inferred 8,689, 8,916 and 9,074 families identical to those in FlyBase, respectively, while the intersection with FlyBase for resolved families decreased slightly for DiffMGC$\overset{\approx}{\text{H}}$ (8,682). For resolved complete families, however, the intersection of DiffMGC$\overset{\approx}{\text{H}}$ and FlyBase families is the largest again, with 5,507 families (89% of the FlyBase set), against, 4,323 (70%), 5,231 (85%) and 5,356 (87%) for Oma, ProteinOrtho and Poff respectively.

We also counted the numbers of gene homologies that are classified as *true positive* (TP), *false positive* (FP) and *false negative* (FN) with a procedure described in Appendix C. We then computed the values of *precision* $\left(\frac{\text{TP}}{\text{TP+FP}}\right)$ and *recall* $\left(\frac{\text{TP}}{\text{TP+FN}}\right)$ for the four methods. Our tool DiffMGC$\overset{\approx}{\text{H}}$ had the lowest value for precision (probably due to its unrefined ambiguous families) and the highest value for recall, which seems to be consistent to its high agreement with FlyBase families.

Running times

The running times are dominated by the preprocessing for all tools – the computation of pairwise sequence similarities. This step took more than 200 hours for Oma, which relies on an internal implementation of the Smith-Waterman algorithm, and 30 hours for DiffMGC$\overset{\approx}{\mathrm{H}}$, which uses blast [2]. As for ProteinOrtho and its extension Poff, they took 45 minutes for performing sequence alignments with diamond [9].

Having at hand the results of alignments, Oma spent around 1 hour to output the inferred gene families, while ProteinOrtho and Poff spent 10 minutes. DiffMGC$\overset{\approx}{\mathrm{H}}$ took 1 hour to postprocess alignments and generate ILPs, then less than 10 minutes to solve most of ILPs, totaling approximately 14 hours. Only 4 of the 55 ILPs reached the time limit of 2 hours, within a gap to the optimal solution of around 0.1% for 3 and 11% for 1 of them. Another round of postprocessing spent 5 minutes to generate families from the solver results.

6 Conclusions and Discussion

We devised and implemented a heuristic capping for improving our recently developed pipeline DiffMGC [18] for inferring gene families based on genome rearrangements. In DiffMGC we adopted an optimal capping including all connections between the ends of linear segments to allow all possible $(2p_*)!$ capping-sets in the input of the ILP FF-DCJ-Indel that infers the DiffM pairwise orthologs. However, due to the heavy optimal capping, FF-DCJ-Indel fails in handling a pair of genomes if one or both of them (with the dimension of a fruit fly genome) are distributed in a hundred contigs.

In contrast, the new pipeline DiffMGC$\overset{\approx}{\mathrm{H}}$ adopts a lighter heuristic capping including connections only between linear segments that share gene content leading to a smaller number of capping-sets in the input of the same FF-DCJ-Indel that here infers DiffM$\overset{\approx}{\mathrm{H}}$ pairwise orthologs. Data from experiments show that the heuristic capping can indeed be much lighter than the optimal capping, reducing drastically the search space. In the analysis of 11 *Drosophila* (partially assembled) genomes, the number of distinct capping-sets was reduced, on average, from (unfeasible) 1,014! to less than 305!. Although the latter value is still large, the heuristic capping allows FF-DCJ-Indel to efficiently handle a pair of fruit fly genomes where both of them can be distributed in hundreds or even thousands of contigs. The bottleneck of our pipeline is still the ILP pairwise computations that, despite the gain of heuristic capping, solve instances of an NP-hard problem. However, at least for genomes with the dimension of a fruit fly genome, DiffMGC$\overset{\approx}{\mathrm{H}}$ lifts the limitation of requiring chromosome-level assembled genomes, expanding to a great extent its applicability.

Not only the genomes in contig-level could be analyzed, but also the quality of the inferred orthologies was very good. The quality evaluation was done by adopting the gene families curated by the FlyBase consortium as a benchmark. The analysis based on a smaller dataset of three completely assembled *Drosophila* genomes compared the previous workflow DiffMGC with the new DiffMGC$\overset{\approx}{\mathrm{H}}$ and showed that the gene families inferred by the two pipelines are virtually the same. The heuristic capping in practice did not have a negative impact on the inferred gene families, preserving the original (optimal) orthology relations. A larger experimental study based on 11 *Drosophila* genomes, including partially assembled genomes distributed in several contigs, compared DiffMGC$\overset{\approx}{\mathrm{H}}$ to other genome-scale methods, namely Oma, ProteinOrtho and Poff. Our results showed that DiffMGC$\overset{\approx}{\mathrm{H}}$ was able to infer the highest number of families and complete families in common with FlyBase, and was very close to the top on the number of incomplete resolved families. Indeed, our tool had the highest value for recall, which seems to be consistent to its high agreement with FlyBase families. However, for precision DiffMGC$\widetilde{\widetilde{\text{H}}}$ had the lowest value, probably due to its 877 unrefined ambiguous families, the largest of them including 151 genes.

As a future work we intend to refine our ambiguous families by breaking them into smaller families, so that we can improve our precision without losing in our recall rate. We will also replace BLAST by DIAMOND in our pipeline, bringing its preprocessing running times closer to those of ProteinOrtho and Poff. This will allow us to more efficiently evaluate our tool with datasets including larger genomes. Additionally we intend to do a systematic study testing the speed and the quality performances of the heuristic capping for a range of distinct values of the parameters τ and ϵ, so that we can derive a way to estimate good values for these parameters considering the input datasets.

References

1 Adrian M Altenhoff, Jeremy Levy, Magdalena Zarowiecki, Bartłomiej Tomiczek, Alex Warwick Vesztrocy, Daniel A Dalquen, Steven Müller, Maximilian J Telford, Natasha M Glover, David Dylus, et al. OMA standalone: orthology inference among public and custom genomes and transcriptomes. *Genome Res*, 29(7):1152–1163, 2019.

2 Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3):403–410, 1990.

3 Sébastien Angibaud, Guillaume Fertin, Irena Rusu, Annelyse Thévenin, and Stéphane Vialette. On the approximability of comparing genomes with duplicates. *J Graph Algo App*, 13(1):19–53, 2009. `doi:10.7155/jgaa.00175`.

4 Anne Bergeron, Julia Mixtacki, and Jens Stoye. A unifying view of genome rearrangements. In *Proc. of WABI*, volume 4175 of *Lecture Notes in Bioinformatics*, pages 163–173, 2006. `doi:10.1007/11851561_16`.

5 Leonard Bohnenkämper, Marília D. V. Braga, Daniel Doerr, and Jens Stoye. Computing the rearrangement distance of natural genomes. *J Comput Biol*, 28(4):410–431, 2021. `doi:10.1089/cmb.2020.0434`.

6 Marília D. V. Braga, Cedric Chauve, Daniel Doerr, Katharina Jahn, Jens Stoye, Annelyse Thévenin, and Roland Wittler. The potential of family-free genome comparison. In C. Chauve, N. El-Mabrouk, and E. Tannier, editors, *Models and Algorithms for Genome Evolution*, volume 19 of *Computational Biology Series*, chapter 13, pages 287–307. Springer Verlag, Berlin, 2013. `doi:10.1007/978-1-4471-5298-9_13`.

7 Marília D. V. Braga, Eyla Willing, and Jens Stoye. Double cut and join with insertions and deletions. *J Comput Biol*, 18(9):1167–1184, 2011. `doi:10.1089/cmb.2011.0118`.

8 David Bryant. The complexity of calculating exemplar distances. In David Sankoff and Joseph H. Nadeau, editors, *Comparative Genomics*, volume 1 of *Computational Biology Series*, pages 207–211. Kluver Academic Publishers, London, 2000. `doi:10.1007/978-94-011-4309-7_19`.

9 Benjamin Buchfink, Chao Xie, and Daniel H. Huson. Fast and sensitive protein alignment using DIAMOND. *Nat Methods*, 12:59–60, 2015.

10 C. Dessimoz, G. Cannarozzi, M. Gil, D. Margadant, A. C. J. Roth, A. Schneider, and G. H. Gonnet. OMA, a comprehensive, automated project for the identification of orthologs from complete genome data: introduction and first achievements. In *Proc. of RECOMB-CG*, volume 3678 of *Lecture Notes in Bioinformatics*, pages 61–72, 2005.

11 Daniel Doerr, Pedro Feijão, and Jens Stoye. Family-free genome comparison. In João C. Setubal, Jens Stoye, and Peter F. Stadler, editors, *Comparative Genomics: Methods and Protocols*, volume 1704 of *Methods in Molecular Biology*, pages 331–342. Springer Nature, New York, 2018. `doi:10.1007/978-1-4939-7463-4_12`.

12 P. Hall. On representatives of subsets. *Journal of the London Mathematical Society*, s1-10(1):26–30, 1935.

13 Sridhar Hannenhalli and Pavel A. Pevzner. Transforming men into mice (polynomial algorithm for genomic distance problem). In *Proc. of FOCS*, pages 581–592, 1995. doi:10.1109/SFCS.1995.492588.

14 Marcus Lechner, Sven Findeiß, Lydia Steiner, Manja Marz, Peter F. Stadler, and Sonja J. Prohaska. Proteinortho: Detection of (co-)orthologs in large-scale analysis. *BMC Bioinform*, 12(124), 2011.

15 Marcus Lechner, Maribel Hernandez-Rosales, Daniel Doerr, Nicolas Wieseke, Annelyse Thévenin, Jens Stoye, Roland K. Hartmann, Sonja J. Prohaska, and Peter F. Stadler. Orthology detection combining clustering and synteny for very large datasets. *PLoS One*, 9(8:e105015), 2014.

16 Fábio V. Martinez, Pedro Feijao, Marília D. V. Braga, and Jens Stoye. On the family-free DCJ distance and similarity. *Algorithms Mol Biol*, 13(10), 2015. doi:10.1186/s13015-015-0041-9.

17 Alexander C. J. Roth, Gaston H. Gonnet, and Christophe Dessimoz. Algorithm of OMA for large-scale orthology inference. *BMC Bioinform*, 9(518), 2008.

18 Diego P. Rubert, Daniel Doerr, and Marília D. V. Braga. The potential of family-free rearrangements towards gene orthology inference. *J Bioinform Comput Biol*, 19(6):2140014, 2021. doi:10.1142/S021972002140014X.

19 Diego P. Rubert, Fábio V. Martinez, and Marília D. V. Braga. Natural Family-Free Genomic Distance. *Algorithms Mol Biol*, 16(4), 2021. doi:10.1186/s13015-021-00183-8.

20 David Sankoff. Genome rearrangement with gene families. *Bioinformatics*, 15(11):909–917, 1999. doi:10.1093/bioinformatics/15.11.909.

21 Mingfu Shao, Yu Lin, and Bernard Moret. An exact algorithm to compute the double-cut-and-join distance for genomes with duplicate genes. *J Comput Biol*, 22(5):425–435, 2015. doi:10.1089/cmb.2014.0096.

22 Guanqun Shi, Liqing Zhang, and Tao Jiang. MSOAR 2.0: Incorporating tandem duplications into ortholog assignment based on genome rearrangement. *BMC Bioinform*, 11(10), 2010.

23 Tamir Tassa. Finding all maximally-matchable edges in a bipartite graph. *Theoretical Computer Science*, 423:50–58, 2012.

24 Sophia Yancopoulos, Oliver Attie, and Richard Friedberg. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics*, 21(16):3340–3346, 2005. doi:10.1093/bioinformatics/bti535.

A ILP formulation for family-free DCJ-indel

This ILP was developed in our previous work [19], but there it only considered an optimal capping of a family-free relational graph $FFR(\mathbb{A}, \mathbb{B}, \mathcal{S})$. It is an adaptation of the ILP for computing the DCJ-indel distance of family-based natural genomes, by Bohnenkämper *et al.* [5], which is itself an extension of the ILP for computing the DCJ distance of family-based balanced genomes, by Shao *et al.* [21].

Given a valid capping θ (that here can be $\theta_\star$ or $\theta_\approx$), the general idea is searching for a sibling-set that, together with a capping-set, induces an optimal consistent capped decomposition of the capped diagram $\theta(FFR(A, B, \mathcal{S})) = (V, E)$, where $V = V_\gamma^{\mathbb{A}} \cup V_\gamma^{\mathbb{B}} \cup \hat{\theta}(\mathbb{A}) \cup \hat{\theta}(\mathbb{B})$ and the set of edges comprises all disjoint sets of distinct edge types: $E = E_\gamma \cup E_\theta \cup E_{\text{adj}}^{\mathbb{A}} \cup E_{\text{adj}}^{\mathbb{B}} \cup E_{\text{id}}^{\mathbb{A}} \cup E_{\text{id}}^{\mathbb{B}}$. Therefore the same ILP formulation (shown in Algorithm 2) computes either $\text{D}\textsc{iff}(A, B, \mathcal{S})$ (with $\theta_\star$) or $\widetilde{\text{D}\textsc{iff}}(A, B, \mathcal{S})$ (with $\theta_\approx$). A particular feature of this ILP when compared to those from [5] and [21] is that its search space includes all sibling-sets, of any size.

For capturing the properties required for computing the best DCJ-indel weighted cost, whose details can be found in [18, 19], the ILP is distributed in three main parts. Counting indel-free cycles (those without indel edges) makes up the first part, depicted in constraints

Algorithm 2 FF-DCJ-INDEL: ILP for computing the best DCJ-indel weighted cost.

Input: A family-free relational graph $\mathit{FFR}(\mathbb{A},\mathbb{B},\mathcal{S})$ with a valid capping $\theta_\star$ or $\theta_\approx$.

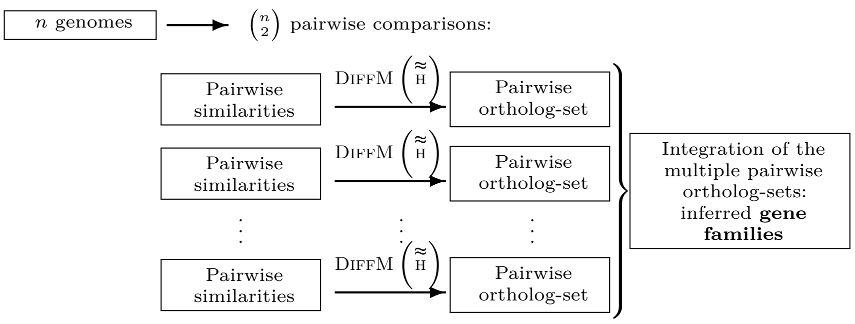
$$\min \quad p + \sum_{e\in E_\gamma} x_e - \sum_{1\le i\le |V|} z_i + \sum_{k\in K} s_k + \frac{1}{2}\sum_{e\in E} t_e - \frac{1}{2}\sum_{e\in E_\gamma} w_e x_e + \sum_{e\in E_{\text{id}}} w_e x_e$$

$$\begin{array}{lrcll}
\text{s. t.} & x_e & = & 1 & \forall\, e \in E^{\mathbb{A}}_{\text{adj}} \cup E^{\mathbb{B}}_{\text{adj}} \quad \text{(C.01)}\\
& \sum_{uv\in E} x_{uv} & = & 2 & \forall\, u \in V \quad \text{(C.02)}\\
& x_e & = & x_d & \forall\, e,d \in E_\gamma,\ e,d \text{ are siblings} \quad \text{(C.03)}\\
& y_i & \le & y_j + i(1 - x_{v_iv_j}) & \forall\, v_iv_j \in E \quad \text{(C.04)}\\
& y_j & \le & y_i + j(1 - x_{v_iv_j}) & \\
& y_i & \le & i(1 - x_{v_iv_j}) & \forall\, v_iv_j \in E^{\mathbb{A}}_{\text{id}} \cup E^{\mathbb{B}}_{\text{id}} \quad \text{(C.05)}\\
& y_j & \le & j(1 - x_{v_iv_j}) & \\
& iz_i & \le & y_i & \forall\, 1 \le i \le |V| \quad \text{(C.06)}\\
& r_v & \le & 1 - x_{uv} & \forall\, uv \in E^{\mathbb{A}}_{\text{id}} \quad \text{(C.07)}\\
& r_{v'} & \ge & x_{u'v'} & \forall\, u'v' \in E^{\mathbb{B}}_{\text{id}}\\
& t_{uv} & \ge & r_v - r_u - (1 - x_{uv}) & \forall\, uv \in E \quad \text{(C.08)}\\
& t_{uv} & \ge & r_u - r_v - (1 - x_{uv}) & \\
& \sum_{d\in E^{\mathbb{A}}_{\text{id}},\ d\cap e\neq\emptyset} x_d - t_e & \ge & 0 & \forall\, e \in E^{\mathbb{A}}_{\text{adj}} \quad \text{(C.09)}\\
& t_e & = & 0 & \forall\, e \in E \setminus E^{\mathbb{A}}_{\text{adj}} \quad \text{(C.10)}\\
& \sum_{e\in E^k_{\text{id}}} x_e - |k| & \le & s_k & \forall\, k \in K \quad \text{(C.11)}\\
\text{and} & x_e & \in & \{0,1\} & \forall\, e \in E \quad \text{(D.01)}\\
& 0 & \le & y_i \ \le\ i & \forall\, 1 \le i \le |V| \quad \text{(D.02)}\\
& z_i & \in & \{0,1\} & \forall\, 1 \le i \le |V| \quad \text{(D.03)}\\
& r_v & \in & \{0,1\} & \forall\, v \in V \quad \text{(D.04)}\\
& t_e & \in & \{0,1\} & \forall\, e \in E \quad \text{(D.05)}\\
& s_k & \in & \{0,1\} & \forall\, k \in K \quad \text{(D.06)}\\
& p & = & p_* \text{ (optimal capping)} \ \text{ or } & p_\approx \text{ (heuristic capping)} \quad \text{(D.07)}
\end{array}$$

(C.01)–(C.06), variables and domains (D.01)–(D.03). The second part is for counting transitions (paths between an indel edge in $\mathbb{A}$ and an indel edge in $\mathbb{B}$, described in constraints (C.07)–(C.10), variables and domains (D.04)–(D.05). The last part describes how to count the number of circular singletons (circular chromosomes exclusively composed of indel and adjacency edges) with constraint (C.11), variable and domain (D.06). The objective function of our ILP minimizes the size of the sibling-set (that is twice the size of the ortholog-set), with sum over variables x_e, the number of circular singletons, calculated by the sum over variables s_k, half the overall number of transitions in indel-enclosing (non-singletons) cycles, calculated by the sum over variables t_e, and the weight of all indel edges in the decomposition, given by the sum over their weights $w_e x_e$ for all $e \in E_{\text{id}}$, while maximizing both the number of indel-free cycles, counted by the sum over variables z_i, and half of the weight of the sibling-set. Note that the minimization is not affected by constant p that corresponds to p_* when the graph is optimally capped or to $p_\approx$ when the graph is heuristically capped.

B Pipeline description

Both the previous DiffMGC with optimal capping and the new DiffMGC$\overset{\approx}{\text{H}}$ with heuristic capping can be summarized in the following pipeline. Given a set of n genomes, for all pairwise comparisons gene similarities and ortholog-sets are computed with the ILP FF-DCJ-Indel solving either DiffM (for DiffMGC) or DiffM$\overset{\approx}{\text{H}}$ (for DiffMGC$\overset{\approx}{\text{H}}$). The resulting pairwise ortholog-sets are then simply integrated into an n-partite graph, and the connected components of this graph are the inferred gene families.



n genomes
$\binom{n}{2}$ pairwise comparisons:
Pairwise similarities
DiffM $\left(\overset{\approx}{\text{H}}\right)$
Pairwise ortholog-set
Integration of the multiple pairwise ortholog-sets: inferred **gene families**

Figure 6 The pipeline of our approach is straightforward: our gene families are the connected components of the n-partite graph derived by the integration of the computed ortholog-sets.

An *ambiguous* family corresponds to a connected component of the n-partite graph that has more than one gene from the same genome. Otherwise we have a *resolved* family, which can be either *complete*, when it contains one gene per genome, or *incomplete* otherwise. Figure 7 illustrates these types of families in a 3-partite graph.

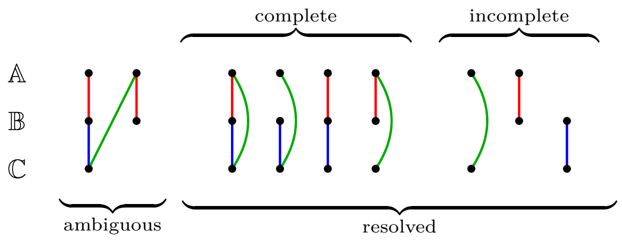


Figure 7 Types of families given by the integration of three ortholog-sets into a 3-partite graph.

C Supplementary information on the analysis of eleven *Drosophilas*

First we show in Figure 8 the distribution of the numbers of resolved FlyBase families (per family size) inferred by each of the four methods.

The numbers of homologies that are classified as *true positive* (TP), *false positive* (FP) and *false negative* (FN) were obtained as follows. Let $\mathcal{H}$ be the subsets of size two of all FlyBase families. For a given method, let $\mathcal{M}$ be the subsets of size two of all inferred families. Then TP is the size of $\mathcal{H} \cap \mathcal{M}$, FN is the size of $\mathcal{H} \setminus \mathcal{M}$ and FP is the size of $\mathcal{M} \setminus \mathcal{H}$. For the four methods we calculated *precision* $\left(\frac{\text{TP}}{\text{TP+FP}}\right)$ and *recall* $\left(\frac{\text{TP}}{\text{TP+FN}}\right)$. The results (Fig. 9) show that DiffMGC$\overset{\approx}{\text{H}}$ (with 877 ambiguous families, the largest of size 151) had the lowest precision and the highest recall, which is consistent with its agreement with FlyBase.

Finally, to give an idea of how many families the four methods have in common, we show the pairwise intersections in Table 1.

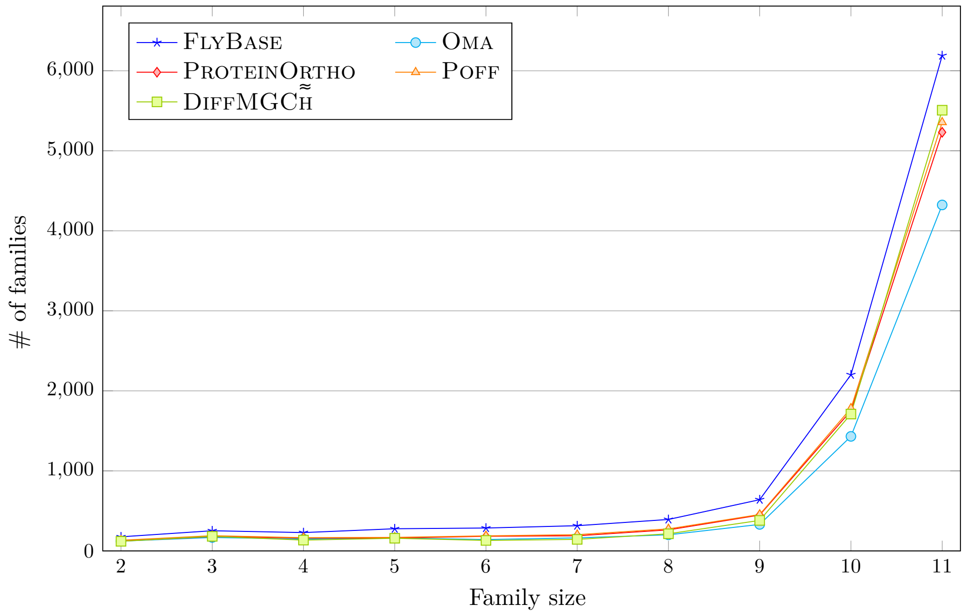


Figure 8 Distribution of resolved families in common with FlyBase families for eleven *Drosophilas*.

Method	TP	FP	FN	precision	recall	F_1-score
Oma	500,500	1,618	114,343	0.997	0.814	0.90
ProteinOrtho	522,462	4,744	92,381	0.991	0.849	0.91
Poff	531,383	3,645	83,460	0.993	0.864	0.92
DiffMGC$\widetilde{\widetilde{\text{H}}}$	561,002	74,801	53,841	0.882	0.912	0.90

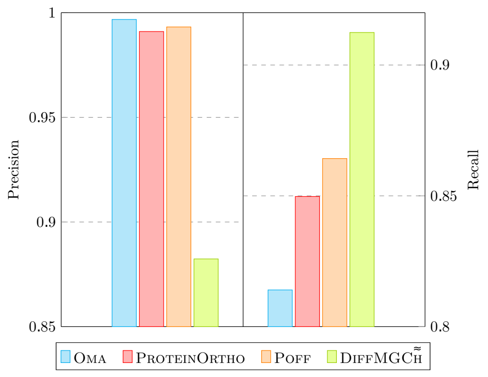


Figure 9 Precision $\left(\frac{\text{TP}}{\text{TP+FP}}\right)$, recall $\left(\frac{\text{TP}}{\text{TP+FN}}\right)$ and their harmonic mean F_1-score for Oma, ProteinOrtho, Poff and DiffMGC$\widetilde{\widetilde{\text{H}}}$, based on the dataset with eleven *Drosophilas*.

Table 1 Comparison of inferred families considering eleven *Drosophila* species. Each cell in the diagonal holds the number of families given by a method and the number of those families that are identical to a FlyBase family. The remaining cells show the number of families in common between two methods and the percentages separated by a dot show the proportions with respect to the total number of families given by the method in the corresponding row and column, respectively.

common resolved incomplete families (FlyBase has 4,769 res. incomplete families)

	DiffMGC$\overset{\approx}{\text{H}}$	Poff	ProteinOrtho	Oma
DiffMGC$\overset{\approx}{\text{H}}$ ∩FlyBase	5,996 100% 3,175 (53%)	3,380 56%.44%	3,209 57%.46%	2,899 48%.30%
Poff ∩FlyBase	- - -	7,707 100% 3,551 (46%)	5,751 75%.82%	4,235 55%.44%
ProteinOrtho ∩FlyBase	- - -	- - -	6,995 100% 3,458 (49%)	4,127 59%.43%
Oma ∩FlyBase	- - -	- - -	- - -	9,673 100% 2,863 (40%)

common resolved complete families (FlyBase has 6,189 res. complete families)

	DiffMGC$\overset{\approx}{\text{H}}$	Poff	ProteinOrtho	Oma
DiffMGC$\overset{\approx}{\text{H}}$ ∩FlyBase	5,834 100% 5,507 (94%)	5,177 89%.88%	5,026 86%.86%	4,257 73%.91%
Poff ∩FlyBase	- - -	5,865 100% 5,356 (91%)	5,501 94%.94%	4,468 76%.95%
ProteinOrtho ∩FlyBase	- - -	- - -	5,835 100% 5,231 (89%)	4,369 75%.93%
Oma ∩FlyBase	- - -	- - -	- - -	4,688 100% 4,328 (92%)